# Economic model for impact of open source software

Asif Khalak*
Department of Mechanical Engineering
Massachusetts Insitute of Technology
Cambridge MA 02139
akhalak@alum.mit.edu

Nov 21 2000

## 1 Introduction

At first glance, the competition between open-source software products and proprietary software products seems puzzling. Simple analogies to other sorts of products, such as consumer goods (e.g. microwave ovens), are counter-intuitive. On one hand, it is unclear how one can successfully demand a price for something which is being given away freely. On the other hand, it does not appear to be sustainable to make open-source products freely available.

The latter point has been partially addressed by Richard Stallman in the GNU manifesto (1985). Stallman points out that the open-source code, even if it is freely distributed, can provide revenue opportunities in complementary areas such as support (this is, in fact, the primary revenue model for RedHat, a major distributor of the open-source operating system, Linux). The primary rationale for open-source according to Stallman, however, is the impetus to work in harmony with other computer users by sharing the rights to software. He suggests that public funds (i.e. a tax) might be appropriate in offseting the expense of making software a public good. Many of Stallman's arguments are written in the hypothetical, since they pre-date the dramatic growth of open-source development, largely associated with the GNU tools, and the Linux operating system. Open-source development was explored further by Eric Raymond in his paper, The Cathedral and the Bazaar (1997). A case-study is presented of an open-source project, fetchmail, led by Raymond. He debunked the myth that open-source software was developed selflessly for the greater good. Rather, the "bazaar" model for software development is proposed in which loosely-connected developers, each acting in their own best interest, can produce high-quality code extremely rapidly. Raymond attributes Linus Torvalds, originator of Linux, with paving the way for bazaar-style development with the Linux kernel. His observations had a dramatic impact, contributing to the decision by Netscape to develop their browser software open-source.

For the purposes of this paper, we assume that open-source generates a software product of same quality to that provided by proprietary development. One of the questions initially raised still holds, however. Namely, whether a software company can survive in the face of competitition from comparable open-source software,

*Current Address: Alphatech, Inc., Burlington MA 01803

which literally gives its product away. Are such companies doomed from the outset? Facing a "new" open-source competitor, would a company be advised to exit the market to avoid losses?

The economic study of goods for which there is no charge (i.e. free goods) has been long studied. Hardin's (1968) described allocation of resources in the context of population growth and the use of natural resources (drawing upon studies dating back to Adam Smith). He described the "Tragedy of the Commons" in which an individual enjoys the full benefit of a free resource, but shares the costs with the rest of society. In this case, there is an incentive to overuse that resource (the Commons) until its ruin. Software markets, however, are unique in the sense that the marginal costs (i.e. the cost associated with an extra unit) are essentially negligible. Ghosh (1998) describes the trade of free goods on the internet in terms of a "cooking pot." That is, contributors donate their software to a common pot, the mixture of which may be freely sampled by anyone. Raymond (1998) proposes that the driving factor encouraging software donations to the internet pot is "noo," a combination of reputation and ego arising from creating and distributing a software solution to a specific need.

Although this previous work provides some arguments concerning the latter question in the first paragraph, they do not address the first question. One general theory of buying behavior, (Howard and Sheth, 1969), categorizes the buying decision process into perceptual and learning subcomponents. In this framework, two possible reasons for choosing a more expensive option of same quality include uncertainty regarding the underlying quality (perceptual component), and unawareness of the product's existence (learning component).

We address the economic interaction of commerical and free products by constructing a mathematical model for the commercial software economy, and using it to explore the effect of an open-source entrant into the market. To be useful in the case of interest, such a model must successfully capture non-standard features such as the dynamics of markets away from equilibrium and actions of parties based upon incomplete information. An *agent-based* economic model (Khalak, 1999) was used, in which modeling occurs on the level of the individual actions of the producers and consumers in the economy. Since the modeling occurs on the level of the individual, its idealizations and limitations are more readily apparent. However, there is no way to analytically solve the model for the results; rather, in every situation, they must be computed by computational simulation.

## 2 Model

The economic model in this case is framed in terms of autonomous agents. That is, individual models for the actions of "buyers" and "sellers" in the economy are used to populate a computer simulation. Such a model specifies the agent inputs, outputs and the relationship of these with its internal state. The simulation runs, and statistics of the agents in the model are tracked to formulate a time-history of the simulated economy. The effects of changing various model parameters can be assessed by comparing simulation runs. The goal of this simulation is not to produce numerically accurate results for use in economic forecasting, but rather as a tool to explore the parameter space of a well-specified model for open-source economics.

The framework of agents has several relevant benefits to the current goals. By using agents, the most important economic assumptions of the model concern the actions of buyers and sellers, rather than assumptions on the level of the entire market. In particular, it is straightforward to specify the information available to the various agents for decision-making.

Some general properties of agent-based systems, however, are still unresolved. For example, general conditions for convergence/scalability and stability of solutions are not available for agent-based systems, except in special cases. Even the simplest such systems (1-D cellular automata) can exhibit complex behaviors which

| | |
|---|---|
| $b_{\text{pref}}$, $b_{\text{ran}}$ | code bank parameters |
| $\alpha_i$ | risk factor for company $i$ |
| $c_i$ | fixed costs for company $i$ |
| $p_j$ | maximum price that user $j$ will pay |
| $P_{i,n}$ | price company $i$ charges at time $n$ |
| $Q_{i,n}$ | quantity sold by company $i$ at time $n$ |
| $AD_{i,n}$ | profits/advertising budget of $i$ at $n$ |

Table 1: List of parameters (top) and state variables (bottom) in software economic model

### 2.3 Code Bank

The code bank acts as a proxy for user buying decisions to separate the effects of price from the effects of brand name. It generates a list of codes whose order depends upon the brand name and upon random effects which are different for each user. The brand name provides the market share and advertising of the code. The list order is in decreasing 'visibility' of the codes in terms of market share, advertising, and the random effects. This visibility list represents the order in which users encounter/prefer codes.

For each user, $j$, the code bank computes the visibility index, $VI_j$, for the code offered by company $i$ at time step $n$, in the following manner:

$$VI_j = (1 - b_{\text{ran}}) \left[ (1 - b_{\text{pref}}) MS_i + b_{\text{pref}} AD_i \right] + b_{\text{ran}} X_j \tag{1}$$

where $b_{\text{pref}}$ sets the relative weight of advertising to market share, $b_{\text{ran}}$ sets the relative weight of random effects to deterministic effects, $MS_i$ is fraction of market share, $AD_i$ is fraction of total advertising, and $X_j$ is a random number between 0 and 1 (uncorrelated between users, with a uniform distribution).

### 2.4 Users

Each user, $j$, has a unique price ceiling, $p_j$, which remains unchanged throughout the simulation. The user queries the code bank and is offered a code (in order of their visibility index, $VI_j$). If the code offered is below the price ceiling, then the user buys a single copy of that code. If not, the user will continue to query the code bank until a code is found which is offered below $p_j$. If no such code is found, then the user does not buy any code (presumably the money is spent on other things.) As such, this model addresses markets in which consumers make their own purchasing decisions. For institutional purchasing, a more complicated model may be appropriate.

The distribution of $p_j$, which is kept fixed over the course of a simulation run specifies the demand curve. We consider a piecewise-linear $p_j$ distribution as depicted in Figure 2, in terms of a minimum and maximum price ceiling, $p_{\min}$ and $p_{\max}$. In the current study, $p_{\max}$ is normalized to 3.

### 2.5 OSS Developers

The behavior of the OSS developers is very simple. Either they offer their code for free, or they do not to offer it at all. Thus, there are two cases. The model explores the transient effect of the OSS developers placing their

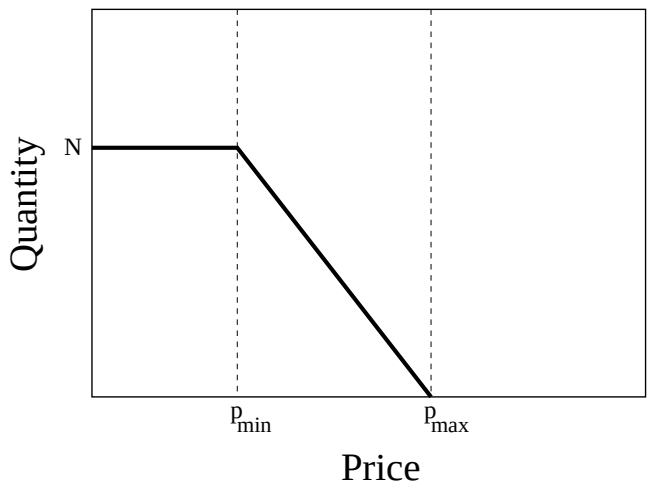



Figure 2: Schematic of demand distribution. Here, $N$ is the number of users in the model, $p_{\min}$ is the minimum price ceiling for everyone, and $p_{\max}$ is the maximum price ceiling. In the current case, the curve is normalized such that $p_{\max} = 3$.

product on the market (or removing it) after a specified time.

### 2.6 Companies

At each step, each company has to send advertising money to the code bank, and decide upon a price to charge for their code. Although they would like to act in a manner to maximize their profits, they are given incomplete information with which to do this. In particular, they do not know *a priori* what the demand for their products will be. After the second time step, they use the last two steps to decide their course of action. Labeling the current time step $n$, each company has the following information:

$$Q_{n-2}, P_{n-2}, Q_{n-1}, P_{n-1}$$

where $Q$ is the quantity sold and $P$ is the price charged.

The profits from the previous time step are simply the revenue minus the costs.All of these profits are assumed to go to the code bank for advertising. However, if profits were negative in the previous time step, the advertising is zero (not negative). The advertising budget, $AD_{n,i}$ can be expressed as

$$AD_{n,i} = \max(0, Q_{n-1}P_{n-1} - c_i) \tag{2}$$

where $c_i$ is the fixed cost of operation.

The price for the code must also be determined. From the data from the previous two time steps, the company could estimate their revenue at time $n$, depending upon the price they charge, assuming a linear demand curve and fitting it to the previous two points. An optimal price could then be computed. However, this assumption of linearity involves a significant risk. It is a justifiable guess for small changes in price, using smoothness, but becomes riskier as the proposed price change increases.

One way to make a statistically robust decision is to use the previous information to determine the *direction* of optimal price movement, but actually change the price by a small, fixed increment. This rule can be expressed as follows. For a given company at time $k$, the new price, $P_k$, is set by

$$P_k = \begin{cases} P_{k-1}(1+\alpha_i), & \left.\frac{\partial R}{\partial P}\right|_{\mathrm{est}} > 0 \\ P_{k-1}(1-\alpha_i), & \left.\frac{\partial R}{\partial P}\right|_{\mathrm{est}} < 0 \end{cases}$$

where $R$ is the revenue, and $\alpha_i$ sets the incremental increase/decrease of the price for company $i$. We term this the 'risk factor' for company $i$. The estimated sensitivity of revenue to price, $\partial R/\partial P$, can be expressed as follows:

$$\left.\frac{\partial R}{\partial P}\right|_{\mathrm{est,k}} = \frac{Q_{k-1}P_{k-1} - Q_{k-2}P_{k-2}}{P_{k-1} - P_{k-2}}$$

# 3 Scope of Study

For the results of simultations of the above model to be useful, one must first establish that these results are repeatable, and that we explore the most relevant part of the parameter space. In the current study, the parameters, $c_i$ and $\alpha_i$ were kept fixed, while the other parameters were explored. The focus is upon the effects of the code bank parameters ($b_{\mathrm{pref}}$ and $b_{\mathrm{ran}}$), and the market demand structure (i.e. price ceiling distribution, $p_j$).

The fixed costs, $c_i$, were kept the same for all the companies, as well as the risk factor, $\alpha_i$. The risk factor was set at 1%, which is small enough so small changes in risk factor did not significantly affect results. Fixed costs were set at $1/4$ of one company's share of the total possible revenue in the market. Although this value is somewhat arbitrary, changes in the level fixed costs did not drastically change the results. Typically, the simulations were performed with 2500 users and 10 companies.

The initial conditions were set such that all companies began at the same point and randomly moved prices in the first round (since they did not have enough information to estimate the sensitivity of revenue to price changes).

## 3.1 Repeatability

Since the model has a stochastic component, $X_j$ in equation (1), the outcome of each simulation differs from run to run. However, we wish to characterize the simulations in terms of statistics which are dependent upon the system parameters, but have a small variance from run to run. When the variance is small enough, it is appropriate to use the *ensemble average* of the data to characterize it, and neglect the run to run variations.

To test this, we ran the simulation several times, tracking the market share of the top two companies. Figure 3 shows the market share of the dominant, $\square$, and second most dominant, $\times$, companies 200 time steps after the start of the model (before the introduction of open-source software). The level of randomness, $b_{\mathrm{ran}}$ is kept constant at 0.25, but several values of $b_{\mathrm{pref}}$ were tested. The distribution of $p_j$ was as in Figure 2 with $p_{\min} = 0.8$ and $p_{\max} = 3.0$. The spread of data at a particular $b_{\mathrm{pref}}$ shows the variation from run to run. This statistic appears to be suitably repeatable. Notice that the choice of statistic is important, too. Since the companies are all identical at the outset, the particular company which winds up being the dominant company is not predictable. Thus, the market share of a specific company in this model might not produce a useful statistic. Although the *market structure* is dependent upon parameters, the exact system evolution is stochastic.

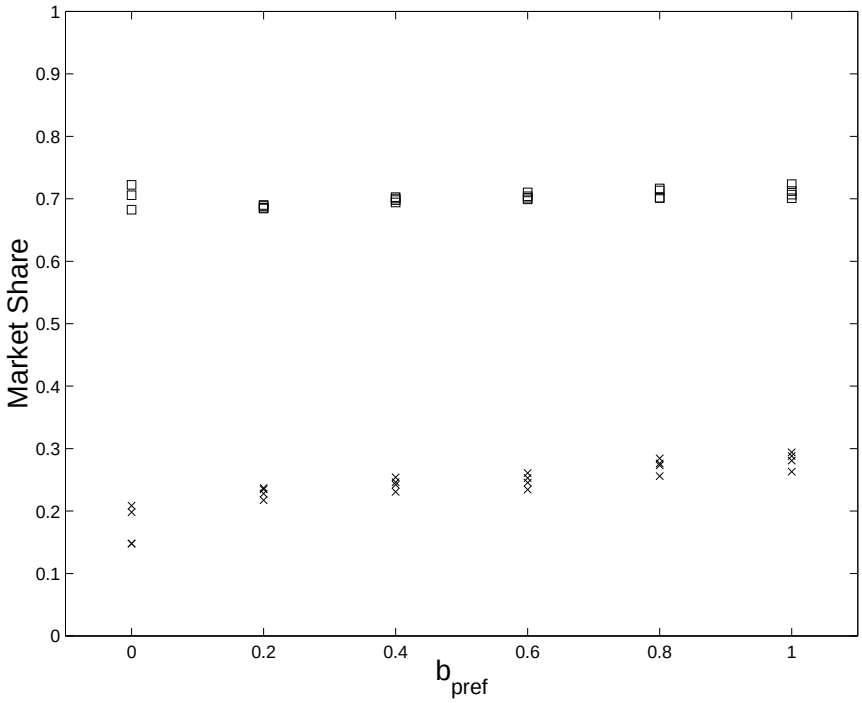



Figure 3: Market share of dominant (□) and second (×) company without open-source software for $b_{\mathrm{ran}} = 0.25$, and various $b_{\mathrm{pref}}$. The spread of values from different runs is small, which indicates the simulation produces repeatable results (despite the stochastic element). Such repeatability suggests that the relative market share is a useful statistic.

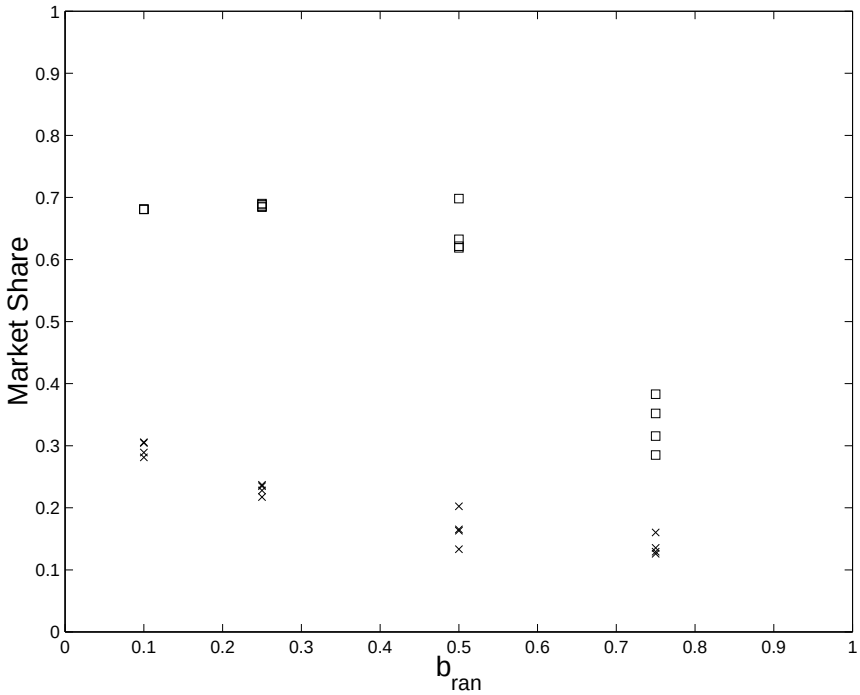



Figure 4: Market share of dominant (□) and second (×) company without free software, after 200 time steps, for $b_{\mathrm{pref}} = 0.2$, and various $b_{\mathrm{ran}}$. A shift occurs between 0.5 and 0.75, where monopolies become untenable because random affects are too great.

# 4 Properties of Baseline Market

Although our primary focus is upon the effects of open-source, the equilibration of the baseline market (without open-source) is of interest as well. Some useful information comes from the market share behavior presented in Figure 3. It appears that in many cases, a dominant company has an effective monopoly, with over 70% of the market. Since the model is initiated with equal properties (and the same behavior) for all companies, the particular company which achieves a monopoly position is essentially random.

From the scoring of the visibility index, $VI$, it is clear that there are gains to "bigness" in this model for software markets. A monopoly has, by definition, a large proportion of the market share which increases its visibility. Further, one expects the monopoly to have a large profit, which goes towards increasing advertising, and further enhancing the visibility index. In terms of the current model, the market leader is established relatively quickly after the start of the simulation (approximately 10-20 timesteps). In real software markets, the benefit of being "first to market" is well appreciated.

While the market share of the second company in Figure 3 appears to have a slight trend with $b_{\mathrm{pref}}$, the market share of the first company seems flat. Varying $b_{\mathrm{ran}}$ while keeping $b_{\mathrm{pref}}$ constant, as in Figure 4, shows that there is a level of $b_{\mathrm{ran}}$, above which a monopoly cannot be sustained. Roughly speaking, as buying habits become less and less predictable, it becomes impossible to capture a dominant part of the market.

Interestingly, in some cases, the market leader does not appear to be invincible. Rather, it is possible to overtake the market leader by undercutting in price, thereby consolidating a significant market share to gain

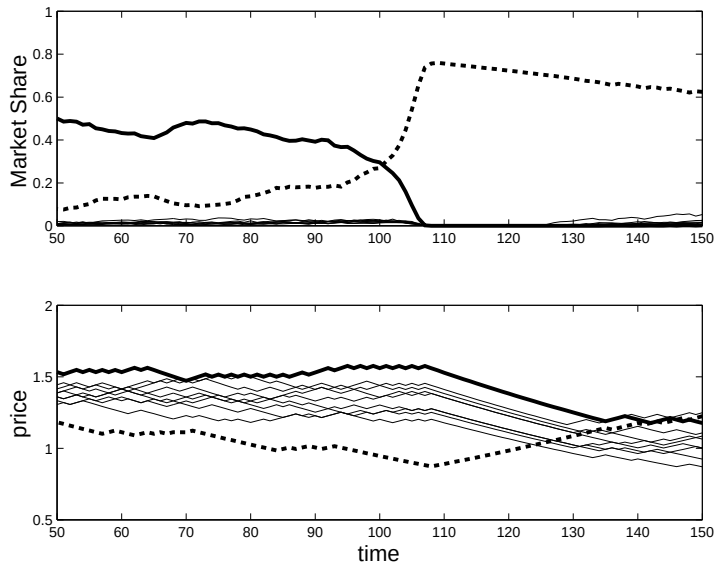



Figure 5: Secondary company overtakes the dominant company by undercutting in price. Time series of software economy, before open-source introduction. One company is initially dominant dominant (solid line), and is able to demand a high price. The secondary company (dashed line) is able to gain market share by undercutting in price, eventually displacing the market leader. In this case, $b_{\text{pref}} = 0.4$ and $b_{\text{ran}} = 0.5$.

visibility, and then subsequently increasing revenues. Although, this strategy is not explicitly coded into the actions of the company agents models (they simply follow the algorithm specified previously), their behavior elicits this mechanism nonetheless. In agent-based systems, this is sometimes termed "emergent" behavior. Figure 5 shows a time-series of one company overtaking another company by this mechanism. Here we consider code bank parameters of $b_{\text{pref}} = 0.4$ and $b_{\text{ran}} = 0.5$, and a demand curve specified by $p_{\min} = 0.2$ and $p_{\max} = 3.0$.

A maps of $(b_{\text{pref}}, b_{\text{ran}})$ are shown in Figure 6 for the case of $b_{\min} = 0.2$ and $b_{\max} = 3$. These corroborate the trends suggested in Figures 4 and 3. The plot is identical for $b_{\min} = 0.8$. That is, above a certain threshold value, $b_{\text{ran}} \approx 0.6$ above which monopolies are not tenable. Below this, however, there is a dominant market leader (at any given time).

## 5 Impact of Open-Source

The open-source community was introduced as follows: after the companies and users were run without open-source software for 200 time steps (to equilibrate the system), open-source software was suddenly introduced at time 200. The subsequent behavior was computed for another 200 time steps.

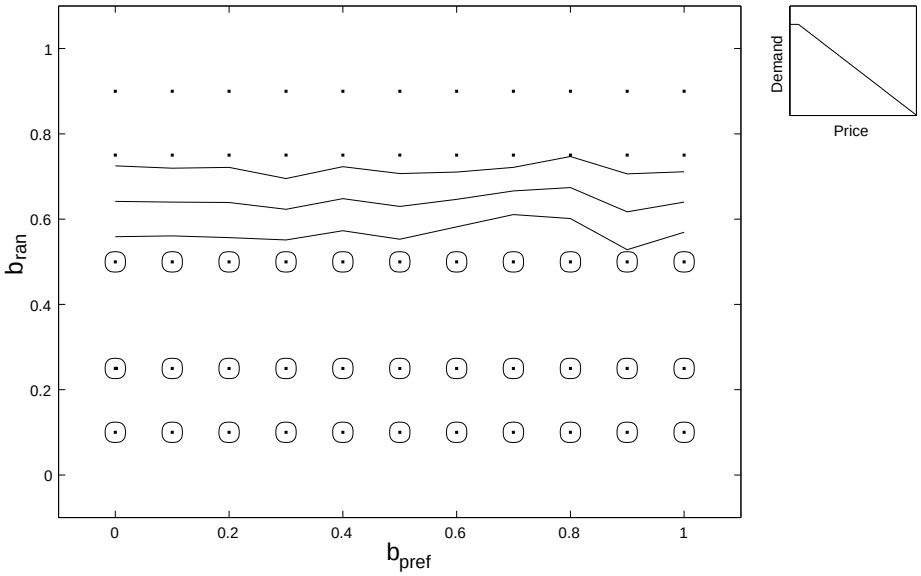



Figure 6: Occurrence of monopolies for various combinations of $b_{\mathrm{pref}}$ and $b_{\mathrm{ran}}$. A circle indicates that there exists a monopoly (Market share $> 75\%$), and a dot indicates otherwise. The contour lines are lines of constant market share of the largest company.

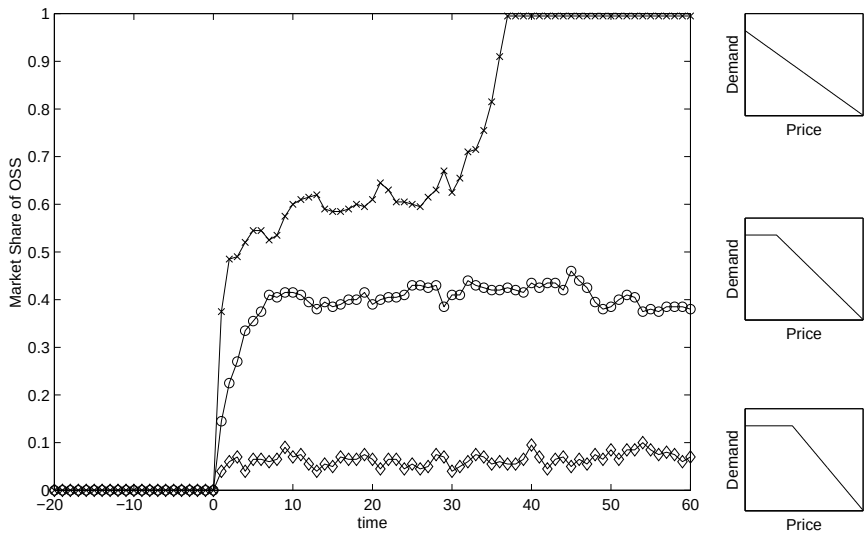



Figure 7: Market share of free products introduced at time=0. Three cases of different demand curves are shown, each with a thumbnail plots of its demand curve to the right.

### 5.1 Effect of Demand Structure

The distribution of $p_j$, determines the demand structure for the market. Figure 7 shows the market share of open-source software introduced for a changing minimum price, $p_{\min}$, as depicted in Figure 2. In eace case, the consumer preferences are constant (i.e. $b_{\text{pref}} = 0.75$ and $b_{\text{ran}} = 0.4$). The three cases considered are $p_{\min} = 0.2$, $p_{\min} = 0.8$, and $p_{\min} = 1.2$, respectively. As the minimum price is increased, the impact of introducing the open-source software is reduced. In the plot, time is measured from the point of the introduction of open-source.

In the first case, of $p_{\min} = 0.2$, the open-source entrant rapidly rises in market share, then slows in growth for about 30 time steps, and finally dominates the market, rapidly gaining 100% of the market share. In the second case, $p_{\min} = 0.8$, the open-source entrant also initially rises rapidly, but plateaus at about 40% of the market. In the third case, the open-source product does not gain a large fraction of the market, but rather remains at less than 10% of the market share.

Another viewpoint, that of corporate revenues, is shown for these three cases in Figure 8. While the previous view displayed the effect of open-source on the users, this shows the effect of open-source on the companies. In this case, before the introduction of open-source, the first two companies control the entire market, with the other companies having essentially a neglible impact. When open-source software is introduced at time t=0, the impact on the corporate revenues depends upon the minimum price, just as in Figure 7. For the case of $p_{\min} = 0.2$, the open-source entry drives all corporate revenues to zero. However, in the case of $p_{\min} = 1.2$, there is relatively less effect upon corporate revenues.

### 5.2 Effect of Purchasing Preferences

In this section, we explore the effects of varying the parameters which represent the purchasing preference, $b_{\text{pref}}$ and $b_{\text{ran}}$, upon the results of introducing an open-source product. That is, varying the relative weights of influences upon users' purchasing decisions. The parameter $b_{\text{pref}}$ indicates the relative preference of market

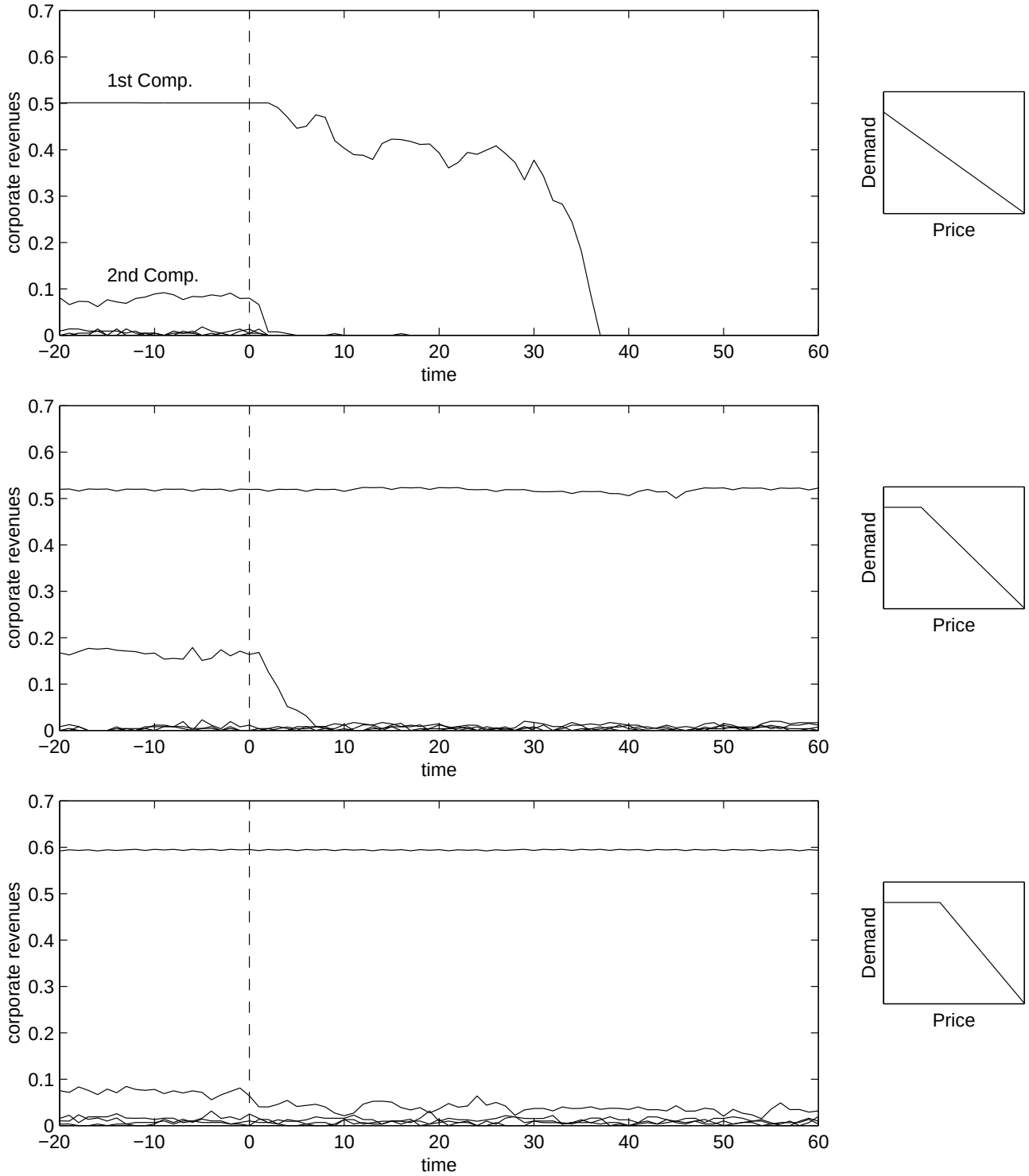



Figure 8: Revenues of commercial products when an open source product is introduced at time=0. Three cases of demand curve are shown, which correspond to the cases of Figure 7. If the $p_{\min}$ is relatively low, open-source is favored and the corporate revenues are adversely affected.

share over advertising in rating product visibility. Further, the parameter $b_{\mathrm{ran}}$ indicates the influence of random effects upon product visibility.

Figure 9 shows the effects of $b_{\mathrm{pref}}$ and $b_{\mathrm{ran}}$ upon the open-source impact. Two cases of demand structure are depicted: $p_{\min} = 0.2$ and $p_{\min} = 0.8$, respectively. Circles are plotted for cases in which open-source gains a monopoly position, and dots are plotted otherwise. Contour lines are plotted for constant open-source market share. Clearly, advertising can counter the effect of open-source when it is highly valued (i.e. low $b_{\mathrm{pref}}$). In such cases, the open source software typically does not appear high enough upon the visibility lists to be purchased. Since the free software must rely upon market share to gain visibility, it never gets it sufficiently to dominate the market. In both cases, there is a region for high $b_{\mathrm{pref}}$ and intermediate $b_{\mathrm{ran}}$ in which conditions for mass adoption of open-source products are favorable.

# 6 Conclusions

Markets exist today in which open-source software competes with proprietary software for the attension of users. This study analyzes the effect of introducing a open-source (i.e. free) software product onto a proprietary market in an idealized setting.

The primary variables characterizing the system were those specifying the market demand, and the relative weights in the user's decision-making between market share, advertising, and random influences. An economic simulation was constructed to study the effects of changing these variables. The simulation was run throughout the parameter space, both without open-source, and following the sudden introduction of open-source.

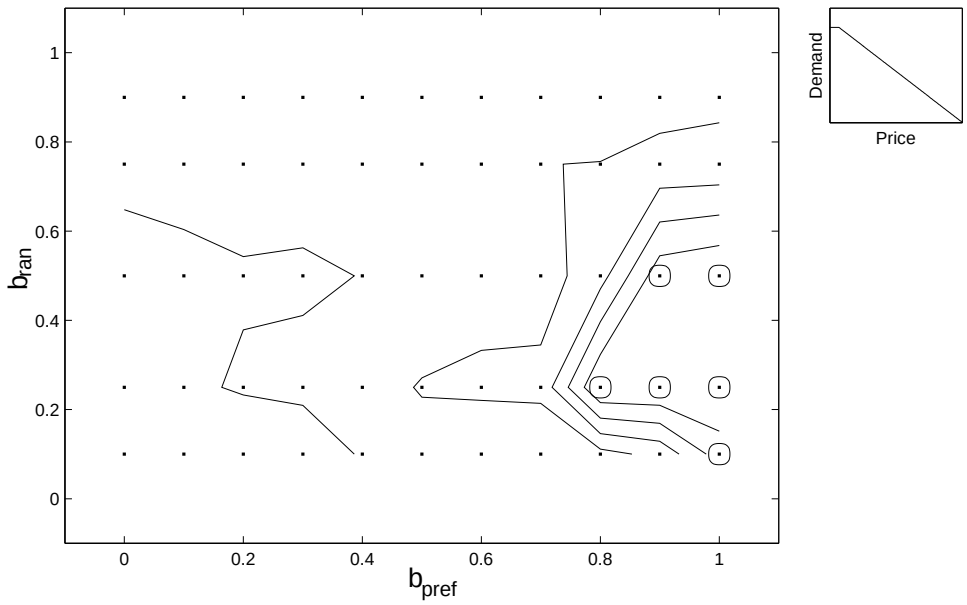
In the baseline case (with only proprietary codes), there was a tendency for monopolies to emerge. Since there are no variable costs, this is not surprising. For cases in which $b_{\mathrm{ran}}$ larger than 0.6, however, there was not a monopoly. The stability of the monopoly position was not guaranteed, since it was possible to displace the market leader by offering a code at a low price and gaining enough market share to displace the monopolist.

The impact of open-source codes was highly dependent upon the value of parameters. For $b_{\mathrm{pref}}$ near 1 (weighted towards market share), and $b_{\mathrm{ran}}$ near 1/2, open-source was able to completely dominate the market (i.e. gain 100% market share). The worst case for open source products was the market in which $b_{\mathrm{pref}}$ and $b_{\mathrm{ran}}$ were both near zero, which is intutitve since it corresponds to the case in which advertising is the sole criterion for purchasing.

Further, the demand structure strongly influences the effect of open-source. In the case that a smaller proportion demanded very low prices, open-source was very successful since it had a ready base of support. On the other hand, as all the users were willing to pay relatively more, open-source had less impact.

The software economy with commercial and free products has several unique and interesting features. The current model shows that even when neglecting the quality and performance of the code, as well as network effects, one obtains an understandable picture of how free and proprietary products can compete with one another. Even if open-source is of equivalent quality and is free of charge, it must still gain a "critical mass" of the market share from those dissatisfied with the product of the market leader (due to high prices in this model) to gain enough inertia to capture the entire market.

A)




B)

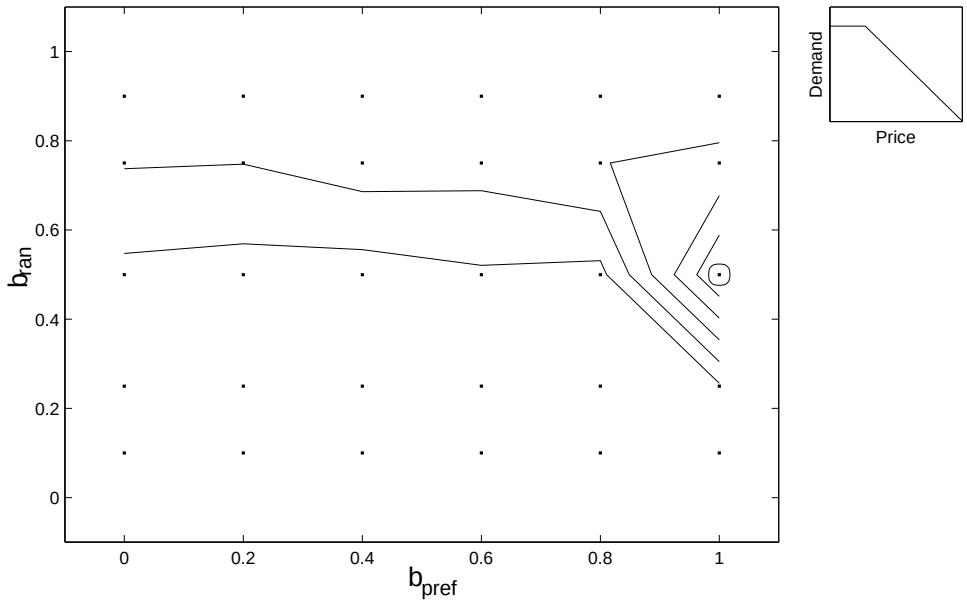



Figure 9: Relative success of introducing open-source software. A circle indicates that open-source software has gained a monopoly. Contour lines of constant open source market share are shown. There is a distinct region at high $b_{\mathrm{pref}}$ and moderate $b_{\mathrm{ran}}$ where free software is likely to dominate.

## Acknowledgements

Discussions with Prof. Seth Lloyd concerning this problem were very very helpful. Further, financial support by the Department of Mechanical Engineering at MIT to present this work is acknowledged.

## References

[1] R. A. Ghosh, Cooking pot markets: an economic model for trade of free goods and services on the internet, `http://dxm.org/tcok/cookingpot/`

[2] G. Hardin "The Tragedy of the Commons," *Science*, 162:1243–1248, 1968.

[3] J. Howard and J. N. Sheth, *The Theory of Buyer Behavior*, Wiley, 1969.

[4] C. Mann, "Programs to the People," *Technology Review*, p.36, Jan-Feb 1999.

[5] A. Khalak "Evolutionary Model for Open Source Software: Economic Impact," *PhD Workshop, Genetic and Evolutionary Computing Conference (GECCO-99)*, Orlando, Fl., June 1999.

[6] E. Raymond, "The Cathedral and the Bazaar," Nov 1997, `http://www.tuxedo.org/~esr/writings/`

[7] E. Raymond, "Homesteading the Noosphere," Apr 1998, `http://www.tuxedo.org/~esr/writings/`

[8] S. Wolfram, "Cellular Automata as Models of Complexity," *Nature*, 311:419–424, 1984